CLAIMS

We claim:

1. A method for capturing information about a structure of a process as it develops, the method comprising:

creating a root task object;

associating code for executing the process with the root task object;

executing the process by invoking the code associated with the root task object; and

whenever a parent task spawns a child task,

creating a task object for the child task;

associating the child task object with a task object of the parent;

associating code for executing the child task with the child task object; and

executing the child task by invoking the code associated with the child task object.

2. The method of claim 1 wherein associating the child task object with the parent task object includes setting a pointer in the child task object to point to the parent task object.

3. The method of claim 2 wherein associating the child task object with the parent task object further includes setting a pointer in the parent task object to point to the child task object.

4. The method of claim 1 wherein associating the child task object with the parent task object includes incrementing a reference counter in the parent task object.

5. The method of claim 4 further comprising decrementing the reference counter in the parent task object when the child task object is deleted.

6. The method of claim 4 further comprising deleting the parent task object when the reference counter reaches a predetermined value.

7. The method of claim 1 further comprising:

associating completion code with a task object; and

executing the completion code when a task associated with the task object completes.

8. The method of claim 7 further comprising:

deleting the task object when the completion code executes.

9. A computer-readable medium containing instructions for performing the method of claim 1.

10. A computer-readable medium having stored thereon a first data structure, the first data structure comprising:

a first data field containing data representing code; and

a second data field containing data representing an association of the first data structure with a second data structure.

11. The first data structure of claim 10 wherein the first data field represents code by pointing to the code.

12. The first data structure of claim 10 wherein the second data field represents the association of the first data structure with the second data structure by pointing to the second data structure.

13. The first data structure of claim 10 wherein the second data structure is the parent of the first data structure.

14. The first data structure of claim 10 wherein the code is code for executing a task.

15. The first data structure of claim 14 further comprising:

a third data field containing data representing a state of the task.

16. The first data structure of claim 15 wherein the state of the task is "pending on object deletion" and wherein the first data structure further comprises:

a fourth data field containing data representing an association of the first data structure with an object on whose deletion the task is pending.

17. The first data structure of claim 15 wherein the state of the task is "pending on task completion" and wherein the first data structure further comprises:

a fourth data field containing data representing a second task on whose completion the task is pending.

18. The first data structure of claim 15 wherein the state of the task is "pending on group de-initialization" and wherein the first data structure further comprises:

a fourth data field containing data representing a group on whose de-initialization the task is pending.

19. The first data structure of claim 14 further comprising:

a third data field containing data representing completion code.

20. The first data structure of claim 10 wherein the code is deletion code.

21. The first data structure of claim 10 further comprising:

a third data field containing data representing an association of the first data structure with a third data structure.

22. The first data structure of claim 21 wherein the third data structure is a child of the first data structure.

23. The first data structure of claim 10 further comprising:

a third data field containing data representing a reference counter.

24. The first data structure of claim 23 wherein the reference counter counts a number of child data structures that are associated with the first data structure.

25. The first data structure of claim 10 further comprising:

a third data field containing data representing a location in source code of a task.

26. The first data structure of claim 10 further comprising:

a third data field containing data representing a resource associated with the first data structure.

27. The computer-readable medium of claim 10 further comprising:

the second data structure, wherein the second data structure comprises:

a third data field containing data representing code; and

a fourth data field containing data representing an association of the second data structure with a third data structure.

28. A method for capturing a structure of an association of resources as it develops, the method comprising:

creating a root resource object; and

whenever a parent resource spawns a child resource,

creating an object for the child resource; and

associating the child resource object with an object of the parent resource.

29. The method of claim 28 wherein associating the child resource object with the parent resource object includes setting a pointer in the child resource object to point to the parent resource object.

30. The method of claim 28 wherein associating the child resource object with the parent resource object further includes setting a pointer in the parent resource object to point to the child resource object.

31. The method of claim 28 wherein associating the child resource object with the parent resource object includes incrementing a reference counter in the parent resource object.

32. The method of claim 31 further comprising decrementing the reference counter in the parent resource object when the child resource object is deleted.

33. The method of claim 31 further comprising deleting the parent resource object when the reference counter reaches a predetermined value.

34. The method of claim 28 further comprising:

associating deletion code with a resource object; and

executing the deletion code when the resource is deleted.

35. A computer-readable medium containing instructions for performing the method of claim 28.

36. A computer-readable medium having stored thereon a data structure comprising a resource object created by the method of claim 28.

37. A computer-readable medium having stored thereon a data structure comprising an association of resource objects created by the method of claim 28.

38. A method for coordinating a task's interaction with an event, the method comprising:

associating a task object with the task;

when the task needs to wait for the event to occur,

suspending execution of the task and associating with the task object a resource object of a resource that causes the event, the association indicating that the task is waiting for the event to occur; and

when the event occurs,

searching the resource object for the association with the task object indicating that the task is waiting for the event to occur and, if found, resuming execution of the task.

39. The method of claim 38 wherein the associating of the task object with the resource object comprises placing a reference to the task object in the resource object.

40. The method of claim 39 wherein the associating of the task object with the resource object further comprises placing a reference to the resource object in the task object.

41. The method of claim 38 further comprising:

when the task causes a second event,

searching the task object for an association with a second task object indicating that a second task is waiting for the second event and, if found, resuming execution of the second task.

42. A computer-readable medium having stored thereon a data structure, the data structure comprising:

a first data field containing data representing members of a group; and

a second data field containing data representing a function for manipulating members of the group.

43. The data structure of claim 42 wherein group members are represented by pointers to the group members.

44. The data structure of claim 42 wherein a group member is a software object.

45. The data structure of claim 42 wherein the function in the second data field is in the set: add member to group, delete member from group, get next group member, apply a second function to group members, search for group member, create hash of key of group member, compare key with key of group member.

46. The data structure of claim 45 wherein a reference counter is associated with a group member, the function is delete member from group, and the function does not delete the member from the group if the member's reference counter does not equal a predetermined value.

47. The data structure of claim 42 further comprising:

a third data field containing data representing a second function for manipulating the group.

48. The data structure of claim 47 wherein the second function is in the set: initialize group, de-initialize group.

49. The data structure of claim 48 wherein the second function is de-initialize group and the second function does not de-initialize the group until the group contains no members and no functions for manipulating members of the group are active.

50. A method for controlling the existence of a first software object, the method comprising:

initializing a reference counter associated with the first software object when the first software object is created;

incrementing the reference counter when a second software object that is a child of the first software object is created;

decrementing the reference counter when the child software object is deleted;

when a request is made to delete the first software object, denying the request unless the reference counter equals a predetermined value; and

when the reference counter equals the predetermined value, deleting the first software object.

51. The method of claim 50 wherein the first software object is associated with a resource in a computing system.

52. The method of claim 51 wherein the resource is in the set: task, section of memory, handle to system-supplied object, application-defined object, and lock.

53. The method of claim 50 wherein the first software object is created and deleted by functions supplied by a creator of the first software object.

54. The method of claim 50 further comprising:

incrementing the reference counter when a cross reference is made between the first software object and another software object; and

decrementing the reference counter when a cross reference made between the first software object and another software object is deleted.

55. The method of claim 50 further comprising:

incrementing the reference counter on an indication that a need has arisen for the first software object; and

decrementing the reference counter on an indication that the need no longer exists.

56. A computer-readable medium containing instructions for performing the method of claim 50.

57. A method for coordinating a task's access to a resource, the method comprising:

creating a lock software object;

associating the lock software object with the resource;

when the task requests access to the resource, if the lock software object is not currently associated with another task, associating the lock software object with the task and granting the task access to the resource; and

when the task indicates that it no longer requires access to the resource, disassociating the lock software object from the task.

58. The method of claim 57 further comprising:

associating a lock level with the lock software object;

associating a lock order with the lock software object; and

when the task requests access to the resource,

examining lock levels of other lock software objects associated with the task, and

denying the request if granting the request would violate the lock order, given the lock levels of the other lock software objects associated with the task.

59. The method of claim 61 wherein functions for associating the lock software object with the task and disassociating the lock software object from the task are supplied by a creator of the task.

60. A computer-readable medium containing instructions for performing the method of claim 61.

61. A computer-readable medium having stored thereon a data structure comprising a lock software object created by the method of claim 57.

62. A system for capturing information about a structure of a process, the process comprising tasks, the system comprising:

a set of task objects, a task object corresponding to a task;

a data structure containing data representing associations among the task objects, the associations reflecting associations among the tasks; and

a set of functions callable by the tasks that build the data structure in response to changes in the associations among the tasks.

63. The system of claim 62 wherein the data structure further contains data representing an association of a resource with a task.

64. The system of claim 63 wherein a resource is in the set: task, section of memory, handle to system-supplied object, application-defined object, and lock.

65. A computer-readable medium containing instructions for providing the system of claim 62.

66. A computer-readable medium having stored thereon a data structure, the data structure comprising:

a first data field containing data representing a count of temporary references existing in a context of a call tree; and

a second data field containing data representing a count of locks held in the context of the call tree.

67. The data structure of claim 66 further comprising:

a third data structure containing data representing an assertion failure function.

68. The data structure of claim 66 further comprising:

a third data structure containing data representing a group of locks held in the context of the call tree.